|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | |  |  | | --- | --- | | **Computer Science (A Level)**  Data Structures Test JDC | **cover_logo_new** | | Please note that you may see slight differences between this paper and the original.  Candidates answer on the Question paper.  **OCR supplied materials:** Additional resources may be supplied with this paper.  **Other materials required:** •   Pencil •   Ruler (cm/mm) | **Duration:** Not set | |  | | |  |

|  |  |  |  |
| --- | --- | --- | --- |
| Candidate forename |  | Candidate surname |  |

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Centre number |  |  |  |  |  | Candidate number |  |  |  |  |

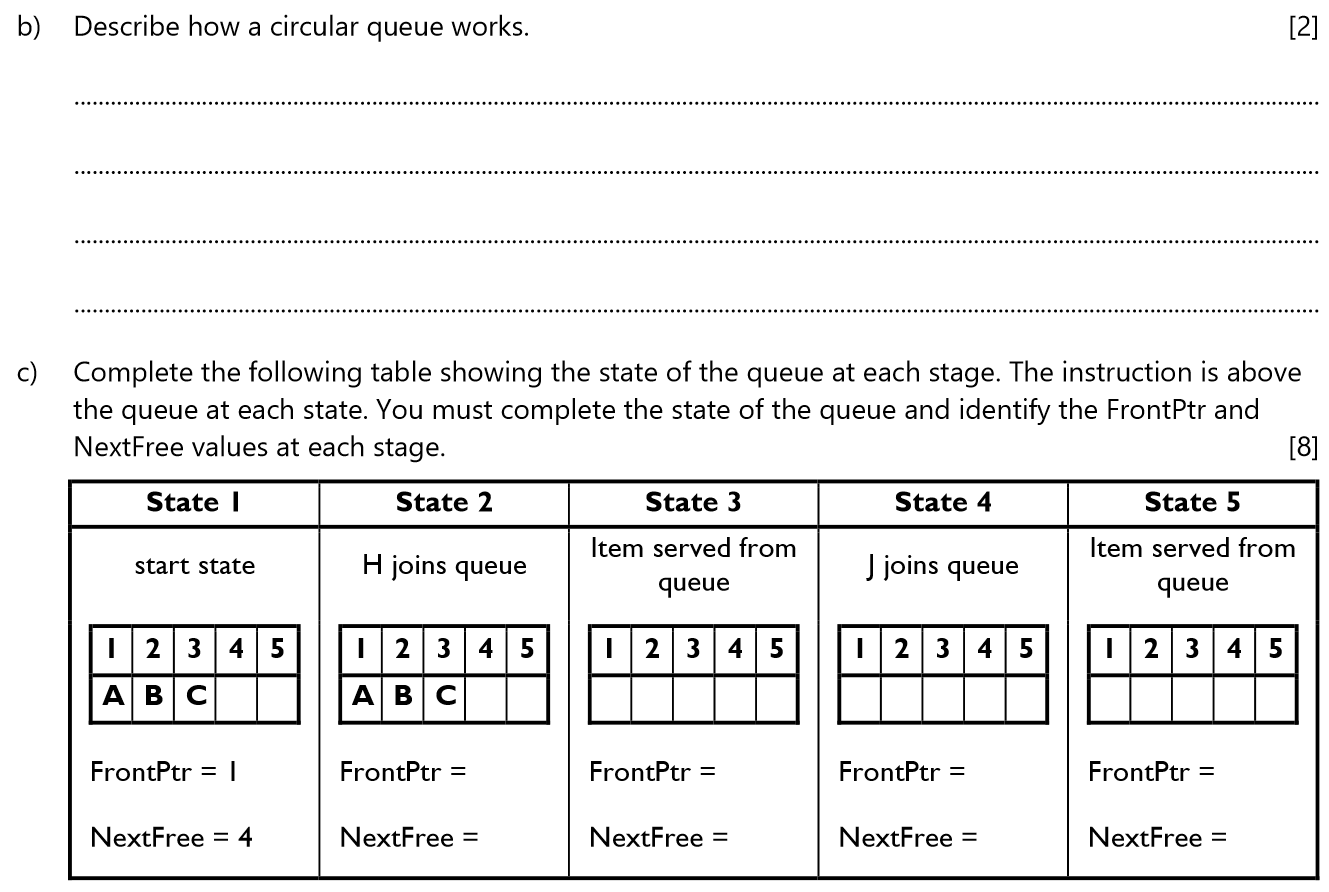
## INSTRUCTIONS TO CANDIDATES

•   Write your name, centre number and candidate number in the boxes above. Please write clearly and in capital letters.  
•   Use black ink. HB pencil may be used for graphs and diagrams only.  
•   Answer **all** the questions, unless your teacher tells you otherwise.  
•   Read each question carefully. Make sure you know what you have to do before starting your answer.  
•   Where space is provided below the question, please write your answer there.  
•   You may use additional paper, or a specific Answer sheet if one is provided, but you must clearly show your candidate number, centre number  
    and question number(s).

## INFORMATION FOR CANDIDATES

•   The quality of written communication is assessed in questions marked with either a pencil or an asterisk. In History and Geography   
    a *Quality of extended response* question is marked with an asterisk, while a pencil is used for questions in which *Spelling, punctuation and  
    grammar and the use of specialist terminology* is assessed.  
•   The number of marks is given in brackets **[ ]** at the end of each question or part question.  
•   The total number of marks for this paper is **46**.  
•   The total number of marks may take into account some 'either/or' question choices.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | |  | |  |  | | --- | --- | |  |  | | **1(a).** | A program stores a queue of mathematical questions to be asked to a user. The questions are asked in the order they are added. Once a question has been asked it cannot be asked again. New questions are continually added to the end of the queue.  The program will use a non-circular queue, questions, (implemented using an array) to store the questions. The pointer, head, stores the index of the first element in the queue. The pointer, tail, stores the index of the last element in the queue.  Fig. 4.1 shows an example of the data in the queue. head is currently 0, tail is currently 4.   |  |  |  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | |  | “2\*3” | “1+4” | “3–1” | “10/2” | “3+6” |  |  |  |  | | Fig. 4.1 | | | | | | | | | |  1. Show the contents of the queue shown in Fig. 4.1, after the following code is run.  |  |  | | --- | --- | |  | add("6+1") |  |  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | --- | | “2\*3” | “1+4” | “3-1” | “10/2” | “3+6” | “6+1” |  |  |  |  |  | | --- | --- | |  | **[2]** |  1. State the values stored in head and tail after the code in **part (i)** has run.  |  |  |  | | --- | --- | --- | | Head | | 0 | | tail | | 5 | |  | **[2]** | | | | |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | |  | |  |  | | --- | --- | |  |  | | **(b).** | Describe why a queue is a suitable structure for this program.   A queue is suitable here as the size, start, and end of the queue are all constantly being  Changed, and the user must be able to remove entries from the start of the queue  But add them to the end, which a normal array cannot do as it has a fixed size      **[3]** | | |
| |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | |  | |  |  | | --- | --- | |  |  | | **2(a).** | A programmer is developing an ordering system for a fast food restaurant. When a member of staff inputs an order, it is added to a linked list for completion by the chefs.  Each element in a linked list has:   |  |  | | --- | --- | | • | a pointer, nodeNo, which gives the number of that node | | • | the order number, orderNo | | • | a pointer, next, that points to the next node in the list |   Fig. 2.1 shows the current contents of the linked list, orders.  pg07_Q_01_150 **Fig. 2.1**  ∅ represents a null pointer.   1. Order 158 has been made, and needs adding to the end of the linked list.   Add the order, 158, to the linked list as shown in Fig. 2.1. Show the contents of the linked list in the following table.  |  |  |  |  | | --- | --- | --- | --- | |  | nodeNo | orderNo | next | |  | 0 | 154 | 1 | |  | 1 | 157 | 2 | |  | 2 | 155 | 3 | |  | 3 | 156 | 4 | |  | 4 | 158 | null |  |  |  | | --- | --- | |  | **[2]** |  1. Order 159 has been made. This order has a high priority and needs to be the second order in the linked list. Add the order, 159, to the original linked list as shown in Fig. 2.1. Show the contents of the linked list in the following table.  |  |  |  |  | | --- | --- | --- | --- | |  | nodeNo | orderNo | next | |  | 0 | 154 | 4 | |  | 1 | 157 | 2 | |  | 2 | 155 | 3 | |  | 3 | 156 | Null | |  | 4 | 159 | 1 |  |  |  | | --- | --- | |  | **[3]** | | |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | |  | |  |  | | --- | --- | |  |  | | **(b).** | The linked list is implemented using a 2D array, theOrders:   |  |  | | --- | --- | | • | Row 0 stores orderNo | | • | Row 1 stores next |   The data now stored in theOrders is shown in Fig. 2.2.  pg08_Q_01_150 **Fig. 2.2**   |  |  | | --- | --- | | theOrders [1,0] would return 1 | | | The following algorithm is written: | | |  | pg08_Q_02_150 |  1. Outline why nodeNo does not need to be stored in the array.   This is because the array already has references starting from 0, meaning you only have to store where in the array the next order is stored  **[1]**   1. Complete the trace table for procedure x, for the data shown in Fig. 2.2.  |  |  |  | | --- | --- | --- | | finished | count | output | | false | 0 |  | |  | 1 | 184 | |  | 2 | 186 | |  | 3 | 185 | | true |  | 187 | |  |  |  | |  |  |  |  |  |  | | --- | --- | |  | **[3]** |  1. Describe the purpose of procedure x.   Procedure x loops through the array and prints out every ordered contained within  It in order    **[2]**   1. A new order, 190, is to be added to theOrders. It needs to be the third element in the list. The current contents of the array are repeated here for reference:  pg10_Q_01_150  Describe how the new order, 190, can be added to the array, so the linked list is read in the correct order, without rearranging the array elements.   to put 190 in the array but be the third element in the list, you must input 190 into  theOrders[0,5], then change the refence of 186 (theOrders[1,1]) to be 5, and the  reference of 190 (theOrders[1,5]) to be 2, so that the list can move to 190 from 186  then back to 185      **[4]** | | |
| |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | |  | |  |  | | --- | --- | |  |  | | **3(a).** | The organisers of an international football competition are planning to use a large electronic score board to display information to spectators in the stadium. The board can display three lines of text of 15 characters each.  The program stores the text to be displayed in an array called Board, so that   * Board(1,1) contains the letter in the top left corner of the display board * Board(3,15) contains the letter in the bottom right corner of the display board.   A module in the program updates the display every time the contents of this array are changed.  State the identifier, number of dimensions and most appropriate data type of the array Board.  Identifier Board  Number of dimensions 2.  Most appropriate data type string[,]  **[3]** | |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | |  | |  |  | | --- | --- | |  |  | | **(b).** | The program contains a module which clears the display using a routine to insert a space in each element of the array using the following algorithm.  Complete this algorithm by filling in the blanks.  p4_01_150  **[3]**  The program contains a module which displays a message at a given position using the algorithm below. For example, DisplayString(“HELLO”,2,1) should display the message “HELLO” on the second row, starting from the first column.  p5_01_150  MID(Message,i,1) returns the character at position i in the string. | | |
| |  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | --- | |  | |  |  | | --- | --- | |  |  | | **4.** | |  | | --- | | A games company has developed a game called Kidz Arrowz. The players throw an arrow at a target board and are awarded different points depending on which circle the arrow lands. Fig. 1 shows the board. | | pg06_Q_01_150 **Fig. 1** |   A computer program is required to keep track of the scores for each competition. The user will enter the number of players, and the name of each player, in that competition to a maximum of 10.   1. The program will then ask for the score of each player in turn. Each competition has 8 rounds, with each player throwing one arrow each round. The program will then display the total score of each player.  The players are declared as a record structure:   record player(string playerName, integer totalScore)  Describe what is meant by a record structure.   a record structure is a method of storing various entries of data, each made up of the same, but individual pieces of information stored in any data type      **[2]**   1. The records for the players are stored in a 1D array.  State why a 1D array is a suitable data structure for the records.   Each player only needs to store two pieces of information, so only one dimension is needed, but the information must be stored in one place  **[1]**   1. Three data structures are arrays, records and stacks.  Identify **one** other data structure.   **Queues [1]** | | |
| |  |  |  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | |  | |  |  | | --- | --- | |  |  | | **5.** | A program stores a queue of mathematical questions to be asked to a user. The questions are asked in the order they are added. Once a question has been asked it cannot be asked again. New questions are continually added to the end of the queue.  The program will use a non-circular queue, questions, (implemented using an array) to store the questions. The pointer, head, stores the index of the first element in the queue. The pointer, tail, stores the index of the last element in the queue.  Complete the following algorithm, to ask the user to input a new question and then either add it to the queue, or report that the queue is full. procedure add()   |  |  | | --- | --- | |  | maxElements = 10  question = input(“please enter your question”)  if tail + 1 >= maxElements  output(“queue is full right now”)  else  questions[tail + 1] = question  tail += 1  end if |   endprocedure   |  |  | | --- | --- | |  | **[4]** | | | |





A circular queue works by storing the front position of the queue, and the next free position, and unless the queue is full, when a new entry in input it is put in at the position in the array referenced by NextFree, and next free is incremented, but if nextfree goes past the end of the array, it is sent back to the start, and the queue then reuses the spaces in the queue that have been removed

**END OF QUESTION paper**